

Механизмы отражения и проксирования

1. Reflection API
2. Механизм проксирования

Отражение (reflection) — способность программы анализировать саму себя.

Рефлексия позволяет:

1. получать информацию о классе, его переменных и методах;
2. получать новый экземпляр класса;
3. получать доступ ко всем переменным и методам;
4. преобразовывать классы одного типа в другой;
5. делать все это во время исполнения программы (динамически, в Runtime).

Недостатки рефлексии:

1. Высокие накладные расходы (низкая производительность)
2. Ограничения безопасности
3. Нарушение принципа инкапсуляции

Class есть у всех объектов в Java.

Class есть у:

- классов, интерфейсов, перечислений;
- примитивов и обёрток над ними;
- массивов;
- void.

```
try {  
    Class<?> carClass = Class.forName("com.package.MyClass");  
} catch (ClassNotFoundException e) {  
    e.printStackTrace();  
}
```

```
MyClass car = new MyClass();  
Class<? extends MyClass> myClass = car.getClass();
```



```
Class<MyClass> myClass = MyClass.class;
```

Исследование модификаторов доступа класса

```
Class c = obj.getClass();  
int mods = c.getModifiers();  
  
if (Modifier.isPublic(mods)) {  
    System.out.println("public");  
}  
  
if (Modifier.isAbstract(mods)) {  
    System.out.println("abstract");  
}  
  
if (Modifier.isFinal(mods)) {  
    System.out.println("final");  
}
```

Метод `getDeclaredFields()`

```
Class<Car> carClass = Car.class;  
Field[] declaredFields = carClass.getDeclaredFields();  
for (Field field : declaredFields) {  
    System.out.println(field);  
}
```

Метод `getDeclaredField()`

```
Class<Car> carClass = Car.class;

try {
    Field horsepowerField = carClass.getDeclaredField("horsepower");
    System.out.println(horsepowerField);

    Field blaBlaField = carClass.getDeclaredField("bla_bla");
} catch (NoSuchFieldException e) {
    e.printStackTrace();
}
```

Метод `getFields()`

```
Class<Car> carClass = Car.class;  
Field[] fields = carClass.getFields();  
for (Field field : fields) {  
    System.out.println(field);  
}
```

Метод getField()

```
Class<Car> carClass = Car.class;

try {
    Field serialNumberField = carClass.getField("serialNumber");
    System.out.println(serialNumberField);

    Field horsepowerField = carClass.getField("horsepower");
} catch (NoSuchFieldException e) {
    e.printStackTrace();
}
```

Получение информации о методах в классе

1. `getDeclaredMethods()`
2. `getDeclaredMethod()`
3. `getMethods()`
4. `getMethod()`
5. `getEnclosingMethod()`

Класс Field предоставляет возможность:

1. получить значение поля, его тип, имя и модификаторы поля
2. получить список аннотаций, класс, в котором объявлено поле и другую информацию
3. установить новое значение в поле, даже если оно объявлено как `private`

Методы класса Field для получения типа переменной:

1. `getByte()`
2. `getShort()`
3. `getInt()`
4. `getLong()`
5. `getFloat()`
6. `getDouble()`
7. `getChar()`
8. `getBoolean()`
9. `get()`

Методы класса Field:

1. `getName()`
2. `getType()`
3. `getModifiers()`

Класс Method предоставляет возможность:

- получить название метода, его модификаторы, тип возвращаемого значения и входящих параметров
- получить аннотации метода, бросаемые исключения и другую информацию
- вызвать метод, даже приватный

Загрузка и динамическое создание экземпляра класса

```
Class c = Class.forName("Test");
```

```
Object obj = c.newInstance();
```

```
Test test = (Test) obj;
```

1. Прокси-класс является `public`, `final` и не `abstract`.
2. Имя прокси-класса по-умолчанию не определено
3. Прокси-класс наследуется от `java.lang.reflect.Proxy`.
4. Прокси-класс реализует все интерфейсы, переданные при создании, в порядке передачи.

Свойства созданного экземпляра прокси-класса:

1. Объект прокси-класса приводим ко всем интерфейсам, переданным в массиве `interfaces`.
2. Статический метод `Proxy.getInvocationHandler` возвращает обработчик вызовов, переданный при создании экземпляра прокси-класса.
3. Класс-обработчик вызовов реализует интерфейс `InvocationHandler`, в котором определен метод `invoke`

1. В чем мотивация использования рефлексии?
2. Зачем нам нужен Proxy?